A HYBRID EVOLUTIONARY ALGORITHM FOR
INTEGRATED PRODUCTION PLANNING AND SCHEDULING PROBLEMS

Lin Lin\(^1,2,\), Xinchang Hao\(^3\), Mitsuo Gen\(^2\) and Katsuhisa Ohno\(^4\)

\(^1\)Dalian University of Technology
Dalian, China
lin@dlut.edu.cn
\(^2\)Fuzzy Logic Systems Institute
Kitakyushu, Japan
gen@flsi.or.jp
\(^3\)Waseda University
Kitakyushu, Japan
janahaoc@gmail.com
\(^4\)Aichi Institute of Technology
Nagoya, Japan
ohno@aitech.ac.jp

ABSTRACT

Integrated production planning and scheduling (IPPS) refers to a manufacturing management process by which raw materials and production capacity are optimally allocated to meet demand. However, most researches presented the different IPPS models with considering the different assumptions under the different manufacturing environment, and proposed the special optimization algorithms. Because the structure difference of the mathematical models, the effectiveness of the proposed algorithm is also different, most IPPS models are difficult to be applied to the practical manufacturing systems.

In this paper, we propose a network modeling way to formulate the IPPS problem into a unified model. In addition, most scheduling models belong to the class of NP-complete problems even when simplifications in comparison to practical problems are introduced. The IPPS transform the original deterministic model to parametric formulations, which makes the problem more complicated. For solving this unified IPPS model, we propose a hybrid evolutionary algorithm (hEA) with combining genetic algorithm (GA) and particle swarm optimization (PSO). Finally, the experiments verify the effectiveness of proposed algorithm, by comparing with different evolutionary approaches for the several test problems.

Key words: Production Planning and Scheduling, Genetic Algorithm, Particle Swarm Optimization, Hybrid Evolutionary Algorithm
1 INTRODUCTION

Integrated production planning and scheduling (IPPS) refers to a manufacturing management process by which raw materials and production capacity are optimally allocated to meet demand. IPPS is especially well-suited to environments where simpler planning methods cannot adequately address complex trade-offs between competing priorities. However, most scheduling problems face inevitable constraints such as due date, capability, transportation cost, set up cost and available resources. We should obtain an effective “flexibility” not only as a response to the real complex environment but also to satisfy all the combinatorial constraints. Thus, how to formulate the complex IPPS problems and find satisfactory solutions play an important role in manufacturing systems.

Lasserre (1992) considered an integrated job-shop planning and scheduling model. A multi-pass decomposition approach alternated between solving: a planning problem for a fixed choice of a sequence of products on the machines; a standard job-shop scheduling problem for a fixed choice of the production plan [1]. Zhang and Yan (2005) also addressed an integrated job-shop production planning and scheduling problem with setup time and batches. It not only considers the setup cost, work-in-process inventory, product demand, and the load of equipment, but also the detailed scheduling constraints. And in order to simultaneously optimize the production plan and the schedule, an improved hybrid genetic algorithm (hGA) is given [2]. Kim et al. (2003) also considered an integrated model of process planning and scheduling in job-shop flexible manufacturing systems, proposed a symbiotic evolutionary algorithm [3]. To increase the flexibility and responsiveness of the job shop in the more competitive market, Li and McMahon (2007) integrated the job-shop production planning and scheduling model, to achieve the global optimization of product development and manufacturing [4]. A simulated annealing-based approach was developed to facilitate the integration and optimization process.

For other manufacturing types, Riane et al. (2001) proposed an IPPS model for serial shops (hybrid flow-shop), and design a decision support system to decomposition into planning and scheduling, and closed loop or feedback mechanism [5]. Yan et al. (2003) proposed a IPPS model for mixed model automobile assembly lines., and proposed a tabu-search-based algorithm to find a combination of a production plan and schedule that are feasible and that approximately optimize the objective function (involving the overproduction and underproduction of finished automobiles, the set-up cost, the idle times of work-cells on the line, the makespan and the load deviations among work-cells) [6].

As discussed above, integrated production planning and scheduling is necessary for an efficient utilization of manufacturing resources. However, in the above literature, researchers considered the different manufacturing environment, proposed the different mathematical models based on the different assumptions, proposed the special optimization algorithms. Because the structure difference of the mathematical models, the effectiveness of the proposed algorithm is also different. Therefore, a unified mathematical model and a general optimization approach are necessary. In addition, most scheduling models belong to the class of NP-complete problems even when simplifications in comparison to practical problems are introduced. The integrated production planning and scheduling transform the original deterministic model to parametric formulations, which makes the problem more complicated. More general, efficient and systematic algorithms are required for recovering feasibility and efficiency with IPPS.

In this paper, we focus on a unified IPPS model design and propose a general optimization approach by evolutionary algorithms. Firstly, we propose a network modeling way to formulate the IPPS problem into a unified model. As we know, an IPPS problem specifies what manufacturing resources and operations/routes are needed to produce a product (a job). In other words, a general IPPS problem can be formulated as a combinatorial optimization problem to decide operation sequence and resources assignment. Then, for solving this unified IPPS model, we propose a hybrid evolutionary algorithm (hEA) with
combining genetic algorithm (GA) and particle swarm optimization (PSO). Finally, the experiments verify the effectiveness of proposed algorithm, by comparing with different evolutionary approaches for the different test problems.

The rest of the paper is organized as follows: In Section 2, we give IPPS modeling explanation, introduce a network modeling way to formulate the problems into a unified model; In Section 3, we propose a hybrid evolutionary algorithm with combining GA and PSO; In Section 4, we demonstrate effectiveness of proposed evolutionary algorithm, by comparing with different evolutionary approaches for the several test problems. Finally, Section 5 rounds up the paper with conclusions.

2 PROBLEM FORMULATION

The Indies of notations are shown as follows:

Indies:

\[ \begin{align*}
    i & \in I \quad \text{job ID} \\
    j & \in J \quad \text{operation ID} \\
    l & \in L \quad \text{material ID} \\
    k & \in K \quad \text{material type ID} \\
    h & \in H \quad \text{resource ID} \\
    u & \in U \quad \text{resource type ID}
\end{align*} \]

2.1 IPPS Modeling

The IPPS problem is how to decide the resources assignment to the production, with considering constraints of resources capabilities and capacities.

Step 1: Resources Checking

The 1st step is enterprise resources checking, define what kinds of resources can be used for manufacturing. There are 2 kinds of definitions (1) material (and semi manufactures) definitions and (2) resource definitions are shown as follows, where constraints included amount, capability and usability etc.

- Material (or semi-manufactures): Material ID, Amount, Usability
- Resource {Resource ID, Amount, Capability, Usability} at time period \( t \).

Material Definitions:

\[ M = \{M_i\} \]

\[ M_i = \{A_k^M, T_k^M\} \]

Material attribute, constraint

Resource Definitions:

\[ R = \{R_h\} \]

\[ R_h = \{A_u^R, T_u^R\} \]

Resource attribute, constraint

Decision variables:

\[ X_i = \{x_{ijh}\} \quad \text{operation resource assignment for each job } i \]

\[ Z_{jj'} = \{z_{jj'h}\} \quad \text{movement resource assignment between operation } j \text{ and operation } j' \]

Step 2: Production Plan Checking

The 2nd step is the checking of production plan, with considering production processes, lot-size, amount and customer requirements etc.
Step 3: Production Process Checking

The 3rd step is the checking of production process for each product (called a job). The job can be formulated as a set of operations, and a production plan can be formulated as a set of jobs.

- **Product (or Job):** Job ID, Name
- **Operations:**
  - BOM (Operations)
  - Operation: Import material requirement, export, Resource requirement
  - Processing Settings: Operation precedence, Operation time

Operation definitions:

\[ N_j = \{t^S_j, t^T_j, p_j, C^I_j, C^E_j\} \quad \text{operation } j \]

\[ t^S_j \quad \text{starting time of operation } j \]

\[ t^T_j \quad \text{ending time of operation } j \]

\[ p_j \quad \text{processing time of operation } j \]

\[ C^I_j = \{M_j, R_j\} \quad \text{import of operation } j \]

\[ C^E_j = \{M_j', R_j\} \quad \text{export of operation } j \]

Operations Relationship:

\[ E_{jj'} = \{(N_j, N_{j'})\} \quad \text{precedence constraint of operation } j \text{ and operation } j' \]

\[ t^-_{jj'} \quad \text{shipment time from operation } j \text{ to operation } j' \]

\[ t^+_{jj'} \quad \text{idle time from operation } j \text{ to operation } j' \]

Decision variable:

\[ Y_{jj'} = \{y_{jj'}\} = \{(0, 1)\}, \forall j, j' \quad 0 \text{ or } 1, \text{ precedence of operation } j \text{ and operation } j' \]

2.2 Network Modeling

Consider a directed network \( G = (N, A) \), consisting of a finite set of nodes \( N = \{1, 2, \ldots, n\} \) and a set of directed arcs \( A = \{(i, j), (k, l), \ldots, (s, t)\} \) joining pairs of nodes in \( N \). Arc \((i, j)\) is said to be incident with nodes \( i \) and \( j \), and is directed from node \( i \) to node \( j \). There are 2 kinds of notations on the directed network: (1) the notations on the node, and (2) the notations on the directed arcs.

For transforming IPPS problem to network model, we define the operations of IPPS problem as a set of nodes \( N \) in the directed network \( G \).

As shown in Fig. 1, for each node (operation \( N_j \)), resources requirement \( \{R_{ib}\} \), start time of operation \( j t^S_j \), end time of operation \( j t^T_j \), processing time of operation \( j p_j \), import of operation \( j C^I_j = \{M_j, R_j\} \), and export of operation \( j C^E_j = \{M_j', R_j\} \) are assigned on the node.

![Figure 1: Illustration Of Node](image-url)
As shown in Fig. 2, a job can be defined as a sub-network. In the sub-network, (1) the directed arc is presenting the operations precedence $E_{jj'}$ for the product design; (2) The variables on the arc are presenting the shipment time $t_{jj'} \text{L}$, idle time $t_{jj'} \text{I}$ and other specialized definitions between operations. The network notations as defined as follows:

**Job Sub-network:**

$$G_i = (N_i, E_i)$$  
sub-network of job $i$

$$N_i = \{N_j\}$$  
operations set for completing job $i$

$$E_i = \{E_{jj'}, t_{jj'} \text{L}, t_{jj'} \text{I}\}$$  
ard from operation $j$ to operation $j'$

![Figure 2: Illustration Of Sub-Network For A Job](image)

As shown in Fig. 3, a production plan can be defined as a directed network $G=\{G_i\}$. The directed network $G$ included all of jobs (with considering lot-size, amount etc.) manufactured. The notation of scheduling graph is shown as following:

**Scheduling Network:**

$$G = \{G_i\}$$  
scheduling network

![Figure 3: Illustration Of Scheduling Network With 3 Job And 11 Operations](image)

### 2.3 Mathematic Modeling

The objectives of network models can be classified as nodes sequencing problems, nodes (or arcs) selection problems, nodes (or arcs) assignment problems, and node grouping problems. The conventional IPPS problems (e.g., JSP based IPPS) and the practical IPPS problems (e.g., iOS/RS) are the combination with nodes sequencing problem and resources assignment problem. Therefore, the IPPS problem decision combined resource assignment $X_i$ and operations precedence $Y_{jj'}$. The objective functions are (1) minimization/maximization of resource parameters (e.g., operation cost), or (2) minimization/maximization of operations parameters (e.g., makespan). For the practical IPPS problems (e.g., integrated scheduling model with multi-plant, AGV dispatching problem [7]) are considering the arcs (movement) assignment problem. The objective functions are (3) minimization/maximization of arc resource parameters (e.g., transportation cost). Sometimes, the special scheduling problems
are considering the node grouping problems (e.g., resources balancing problem). This paper does not consider the node grouping problems.

**Objective Functions:**

\[
\begin{align*}
\text{min/max } f_1(R_h, X_i) & \quad (1) \\
\text{min/max } f_2(p_j, Y_{jj'}) & \quad (2) \\
\text{min/max } f_3(E_{jj'}, Z_{jj'}) & \quad (3)
\end{align*}
\]

**System Constraints:**

\[
\begin{align*}
g_1(N_j, Y_{jj'}) & \geq 0 \quad (4) \\
g_2(E_{jj'}, Y_{jj'}) & \geq 0 \quad (5) \\
g_3(R_h, X_i) & \geq 0 \quad (6) \\
g_4(R_h, Z_{jj'}) & \geq 0 \quad (7)
\end{align*}
\]

**Non-negative Constraints:**

\[
\begin{align*}
X_i = \{x_{jh}\} & \geq 0, \forall i \quad (8) \\
Y_{jj'} = \{y_{jj'}\} = \{(0, 1)\}, \forall j \quad (9) \\
Z_{jj'} = \{x_{jh}\} & \geq 0, \forall j \quad (10)
\end{align*}
\]

The constraints of network models can be classified as nodes precedence constraints, arcs (movement) precedence constraints, and flow (nodes or arcs) capacity constraints. Therefore, the scheduling problems should consider the (4) operation precedence constraints, (5) the transportation precedent constraints, and (6) resource usability constraints, and (7) movement resource usability constraints.

### 3 HYBRID EVOLUTIONARY ALGORITHM

Since the 1960s, there has been an increasing interest in imitating living beings to solve the hard optimization problems. An evolutionary algorithm (EA) is a generic population-based meta-heuristic optimization algorithm. An EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions “live” (see also cost function). Evolution of the population then takes place after the repeated application of the above operators [8]. Handa et al. (2008) gave a comprehensive overview of recent advances of evolutionary computation (EC) studies, as shown in Fig. 4 [9]. EAs differ in the implementation details and the nature of the particular applied problem.

The IPPS problem can be defined as: select suitable manufacturing resources (machines, tools etc.) and sequence the operations so as to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved. So, a general IPPS problem can be formulated as a combinatorial optimization problem to decide operation sequence and resource assignment. An effective EA for scheduling problem has to consider: (1) Effective for the operation sequence, (2) Effective for the resource assignment, and (3) Effective for the combination of the operation sequence and the resource assignment.
3.1 Individual Representation

How to present a solution of the scheduling problem into a chromosome is a key issue for EAs. For evaluating the effectiveness of the different chromosome representation, there are several critical issues are summarized by Gen et al. (2008) [10].

- **Space**: Chromosomes should not require extravagant amounts of memory.
- **Time**: The time complexities of evaluating, recombining, and mutating chromosomes should be small.
- **Feasibility**: All chromosomes, particularly those generated by simple crossover (i.e., one-cut point crossover) and mutation, should represent feasible solutions.
- **Uniqueness**: The mapping from chromosomes to solutions (decoding) may belong to one of the following three cases: 1-to-1 mapping, n-to-1 mapping and 1-to-n mapping. The 1-to-1 mapping is the best one among three cases and 1-to-n mapping is the most undesired one.
- **Heritability**: Offspring of simple crossover (i.e., one-cut point crossover) should represent solutions that combine substructures of their parental solutions.
- **Locality**: A mutated chromosome should usually represent a solution similar to that of its parent.
We need to consider these critical issues carefully when designing an appropriate representation so as to build an effective EA.

3.1.1 Representation for Operation Sequencing

In the past few decades, the following 6 representations for job-shop scheduling problem (JSP, an operation sequencing problem with considering the precedence constraints of operations) have been proposed:

- Operation-based representation (De Jong 1994) [11]
- Job-based representation (Holsapple et al. 1990) [12]
- Preference list-based representation (Croce et al. 1995) [13]
- Priority rule-based representation (Dorndorf and Pesch 1995) [14]
- Completion time-based representation (Yamada and Nakano 1992) [15]
- Random key-based representation (Norman and Bean 1995) [16]

The Flexible Job-shop Scheduling Problem (fJSP) is expanded from the traditional JSP, which possesses wider availability of machines for all the operations (a combinatorial optimization problem considering both of the operation sequence and the resource assignment). The following 4 representations for fJSP have been proposed:

- Parallel machine-based representation (Gen and Cheng, 1997) [8]
- Parallel jobs representation (Gen and Cheng, 1997) [8]
- Operations machines-based representation (Kacem et al. 2002) [17]
- Multistage operation-based representation (Zhang and Gen 2006) [18]

Permutation-based representation is perhaps the most natural representation of operation sequences. Unfortunately because of the existence of precedence constraints, not all the permutations of the operations define feasible sequences. For job shop scheduling problem, Cheng et al. (1996, 1999) applied job-based representation: they name all operations for a job with the same symbol and then interpret them according to the order of occurrence in the sequence of a given chromosome [19, 20]. Gen and Zhang (2006) also applied this representation to advanced scheduling problem [21]. The job-based representation can also be used to represent the operation sequences for the fJSP problem. Each job \( i \) appears in the operation sequence exactly \( n_i \) times to represent its \( n_i \) ordered operations. However, if the operation precedence is more complex than JSP or extend JSP problems, the job-based representation cannot be used directed.

Cheng and Gen (1994) proposed a priority-based representation firstly for solving Resource-constrained Project Scheduling Problem (rcPSP)[22]. This representation encodes a schedule as a sequence of operations and each gene stands for one operation. As we known, a gene in a chromosome is characterized by two factors: locus, i.e., the position of the gene located within the structure of chromosome, and allele, i.e., the value the gene takes. In this encoding method, the position of a gene is used to represent operation ID and its value is used to represent the priority of the operation for constructing a schedule among candidates. A schedule can be uniquely determined from this encoding.

Fig. 3 presents a scheduling network with 3 jobs and 11 operations. Illustration of priority-based representation is shown in Fig. 5. At the beginning, we try to find an operation for the position next to source node S. Operations 1, 5, 8 and 10 are eligible for the position, which can be easily fixed according to adjacent relation among operations. The priorities of them are 10, 5, 8 and 1, respectively. Operation 1 has the highest priority and is put into the schedule. The possible operations next to operation 1 are operations 2 and 3, and possible operations 5, 8 and 10. Because operation 3 has the largest priority value, it is put into the schedule. Then we form the set of operations available for next position and select the one with the highest priority among them. Repeat these steps until we obtain all operations into the schedule, \((N_1, N_5, N_8, N_9, N_6, N_2, N_4, N_7, N_9, N_{10}, N_{11})\).
However, the nature of the priority-based representation is a kind of permutation representations. Generally, this representation will yield illegal offspring when using one-cut point crossover or other simple crossover operators. That means some node’s priority may be duplicated in the offspring. There are several crossover operators proposed for permutation representation, such as partial-mapped crossover (PMX), order crossover (OX), position-based crossover (PX), heuristic crossover, and so on [10]. Norman and Bean (1995) proposed random key-based representation for JSP [16]. In this paper, we combine the random key-based representation for operations sequencing. The example of representation is shown in Fig. 6, and we can obtain the same operations sequence into the schedule, \((N_1 \rightarrow N_2 \rightarrow N_8 \rightarrow N_5 \rightarrow N_6 \rightarrow N_2 \rightarrow N_4 \rightarrow N_7 \rightarrow N_9 \rightarrow N_{10} \rightarrow N_{11})\).

3.1.2 Representation For Resources Assignment

After the operation sequence is fixed, the resources assignment can be formulated as a multi-stage graph problem. For each stage (operation), we decided the state number (which resource should be assigned). This multi-stage graph problem can be solved by dynamic programming. Yang (2001) proposed a GA-based discrete dynamic programming approach for scheduling in FMS environment [23]. However, the IPPS problem is the combination of the operation sequencing (NP-hard problem) and resources assignment. Considering the computation times of the algorithm, and the most of practical IPPS problems are multi-resources assignment, the most of researches combined a state permutation representation into the chromosome [24-26], called multi-stage representation.

For each resource type \(h\), generate a chromosome section by using this state permutation representation. In this representation, the position of a gene is used to represent operation ID, and its value is used to represent the resource id selected. The maximum value is equal to the number of resources \(|R_h|\) of resource type \(h\). In this paper, we use the real number in the state permutation representation.

For example, Fig. 7 presents resource \(R_h\) (machine) assignment for 11 operations. The number of resources \(|R_h|\) is equal to 4. In particular, the operation \(N_3\) is only achievable on a part of the available machines \(M_2\) and \(M_4\). And the operation \(N_4\) is only achievable on a part of the available machines \(M_2, M_3\) and \(M_4\). Illustration of permutation representation is shown in Fig. 8. As the decoding process, we assign the machine \(M_u\) to operation \(j\) by using the following equation:

\[
u^* = \text{FIX}(v_j U_j')\]  \hspace{1cm} (11)

where \(v_j\) is the value of the \(j\)-th gene; \(U_j'\) is the number of available machines for operation \(j\); \(\text{FIX}(x)\) rounds the elements of \(x\) to the nearest integers towards zero.
Evolutionary Operations perform search solution role in the Evolutionary Algorithms (EAs). Using the different evolutionary operators has very large influence on EA performance. EAs use the different evolutionary operations.

Search is one of the more universal problem solving methods for such problems one cannot determine a prior sequence of steps leading to a solution. Search can be performed with either blind strategies or heuristic strategies. Blind search strategies do not use information about the problem domain. Heuristic search strategies use additional information to guide search move along with the best search directions. There are two important issues in search strategies: exploiting the best solution and exploring the search space [27]. Michalewicz (1994) gave a comparison on hillclimbing search, random search and EAs’ genetic search [28]. Hillclimbing is an example of a strategy which exploits the best solution for possible improvement, ignoring the exploration of the search space. Random search is an example of a strategy which explores the search space, ignoring the exploitation of the promising regions of the search space.

Genetic Algorithm (GA) is a class of general purpose search methods combining elements of directed and stochastic search which can produce a remarkable balance between exploration and exploitation of the search space. At the beginning of genetic search, there is a widely random and diverse population and crossover operator tends to perform widespread search for exploring all solution space. As the high fitness solutions develop, the crossover operator provides exploration in the neighborhood of each of them. In other words, what kinds of searches (exploitation or exploration) a crossover performs would be determined by the environment of genetic system (the diversity of population) but not by the operator itself. In addition, simple genetic operators are designed as general purpose search methods (the domain-independent search methods) they perform essentially a blind search and could not guarantee to yield an improved offspring.

Recently, many researchers are focusing on the different type EAs for optimization problems. The main differences are the different evolutionary operations. Ant Colony Optimization (ACO) decodes the individual to solution depending on a priority value $\tau$, called pheromone. The evolutionary operation of ACO is a pheromone updating rule. In each iteration $t$ of the algorithm, all ants (chromosome) have decoded a solution, $\tau$ are updated by means of the following equation:

$$
\tau_{\psi}(t) = \rho \cdot \tau_{\psi}(t - 1) + \Delta \tau_{\psi}
$$

(12)
where \( \tau_{i|\psi} \) represents the sum of the contributions of all ants that used move \((\psi)\) to construct their solution. \( 0 \leq \rho \leq 1 \) is a user-defined parameter, and \( \Delta \tau_{i|\psi} \) represents the sum of the contributions of all ants that used move \((\psi)\) to construct their solution. Zhang and Gen (2009) proposed a parallel hybrid ant colony optimization approach for job-shop scheduling problem [29].

In Particle Swarm Optimization (PSO), the representation code is called particle swarm, and each allele is called a particle. The evolutionary operation of PSO is that particle fly through the problem space by following the current optimum particles. In each iteration \( t \), the algorithm updates positions \([x_i^t]\) and velocities \([v_i^t]\) of the particle as follows:

\[
v_i^t = \omega v_i^{t-1} + \phi_1(g_i^{t-1} - x_i^{t-1}) + \phi_2(l_i^{t-1} - x_i^{t-1}) \tag{13}
\]

\[
x_i^t = x_i^{t-1} + v_i^t \tag{14}
\]

with \( \phi_1 = r_1 a_g \), \( \phi_{12} = r_2 a_l \), \( r_1, r_2 \rightarrow U(0,1) \), \( \omega, a_l, a_g \in \mathbb{R} \). \( l_i^t \) is the best position found so far by the \( i \)th particle and \( g^t \) is the global best position on the swarm. Guo et al. (2006, 2009) proposed PSO algorithms for integrated process planning and scheduling problems [30, 31].

In this paper, we design 2 kinds of evolutionary operations for the manufacturing scheduling problems: exploitation-based evolutionary operation, and exploration-based evolutionary operation. We combine the one-cut point crossover as the exploration-based evolutionary operation. Crossover operates on two chromosomes at a time and generates offspring by combining both chromosomes’ features. One-cut point crossover is a simple way to achieve crossover would be to choose a random cut-point and generate the offspring by combining the segment of one parent to the left of the cut-point with the segment of the other parent to the right of the cut-point.

We combine the evolutionary operation of PSO as the exploitation-based evolutionary operation. In PSO, the particle is replaced by Equation (14) in each iteration \( t \). Different with the operation of PSO, the exploitation-based evolutionary operation replace the chromosome if the generated chromosome is better than the original chromosome. The evolutionary operation process is shown as following steps:

**Step 1:** Calculate velocities \([v_{ij}^t]\) for each gene \( i \) in the chromosome \( j \) in generation \( t \).

\[
v_{ij}^t = \omega v_{ij}^{t-1} + \phi_1(g_{ij}^{t-1} - x_{ij}^{t-1}) + \phi_2(l_{ij}^{t-1} - x_{ij}^{t-1}), \quad \forall i,j \tag{15}
\]

where \( \omega, a_l, a_g \) are parameters of the algorithm, \( \phi_1 = r_1 a_g \), \( \phi_{12} = r_2 a_l \), \( r_1 \) and \( r_2 \) are random numbers between \((0, 1)\). \( g_{ij} \) is the value of gene \( i \) in the best chromosome, and \( l_{ij}^t \) is the value of gene \( i \) in the best chromosome in generation \( t \).

**Step 2:** Generate a new chromosome \( j^* \) with calculating each gene \( i \) in generation \( t \) by following equation:

\[
x_{ij}^{t^*} = x_{ij}^t + v_{ij}^t, \quad \forall i,j \tag{16}
\]

**Step 3:** Evaluate the fitness \( fit_{j^*} \) of the chromosome \( j^* \); if the fitness \( fit_{j^*} \) is better than the fitness \( fit_j \) of chromosome \( j \), replace the chromosome \( j \) by \( v_{ij}^t = [v_{ij}^{t^*}] \).

Selection operation is the evolutionary operation of EA, provides the driving force in a EA. With too much force, a genetic search will be slower than necessary. Typically, a lower selection pressure is indicated at the start of a genetic search in favor of a wide exploration of the search space, while a higher selection pressure is recommended at the end to narrow the search space. The selection directs the genetic search toward promising regions in the search space. During the past two decades, many selection methods have been proposed, examined, and compared. Common selection methods are as roulette wheel selection, (\( \mu + \lambda \))-selection, tournament selection, truncation selection, elitist selection, ranking and scaling, and sharing. In this paper, we combine “Roulette wheel selection” as selection operation. It is to determine selection probability or survival probability for each
chromosome proportional to the fitness value. A model of roulette wheel can be made displaying these probabilities.

Smith and Fogarty (1996) proposed a steady state genetic algorithm with a steady state evolutionary mechanism [32]. They mention the following policies will affect the algorithm:

Deletion Policy: Two standard policies are frequently used with steady state GA’s, namely Delete-Worst and Delete- Oldest. In addition there is the issue of whether a chromosome of the population should be replaced only if it is less fit than the offspring which would replace it (Conditional Replacement) or always (Unconditional Replacement).

Selection Policy: Parental selection is Fitness Proportionate (“Roulette Wheel”). Selection of an individual from the “clutch” to enter the population can use the same mechanism or be deterministic, i.e. the best offspring is always picked.

Clutchsize Policy: i.e. the number of offspring cloned in a single iteration of the algorithm. This will affect the balance between exploitation and exploration in the early stages of the run before the population has converged.

In this paper, we combine proposed exploitation-based evolutionary operation as the deletion policy, combine roulette wheel selection as fitness proportionate, and combine the clutchsize policy for scheduling problems, where the probability of clutchsize

\[ p_s = \frac{\text{clutch size}}{\text{population size}}. \]

3.3 Overall Procedure

The \( P(t) \) and \( C(t) \) are parents and offspring respectively in current generation \( t \), the implementation structure of proposed hEA is described as follows:

```
procedure: hEA for manufacturing scheduling problem
input: scheduling data \((N, E, R, p)\), hEA parameters
output: best solution \(S\)
begin
    \( t \leftarrow 0; \) // \( t \): generation number
    initialize operation sequence section \( P_1(t) \) by random key-based representation;
    initialize resource assignment sections \([P_h(t)]\) by state permutation representations;
    calculate objective \(f(P)\) and evaluate \(\text{eval}(P)\) by decoding process;
    while (not terminating condition) do
        create \(P'(t)\) from \(P(t)\) by exploitation-based evolutionary operation;
        replace \(v_j \leftarrow v_{j'}\), if \(\text{eval}(v_j) < \text{eval}(v_{j'})\) for all \(v_{j'} \in P'(t)\) and \(v_j \in P(t)\);
        create \(C(t)\) from \(P(t)\) by exploration-based evolutionary operation;
        calculate objective \(f(P)\) and evaluate \(\text{eval}(C)\) by decoding process;
        clone best offspring into \(P(t+1)\) by clutchsize policy;
        select \(P(t+1)\) from \(P(t)\) and \(C(t)\) by roulette wheel selection policy;
        \(t \leftarrow t+1;\)
    end
    output the best solution \(S\);
end
```
4 EXPERIMENTAL COMPARISONS AND DISCUSSIONS

To justify the performance of the proposed approach, different EAs are used to evaluate the same test set of problems. The results are compared in terms of solution optimization and convergence rate. All of tests are conducted 30 runs on a machine running on Intel Xeon 2.00GHz CPU and 4 GB of memory.

Kim et al.’s IPPS problems [3] are adopted for the experiments. This IPPS addresses an integrated problem of process planning and scheduling in job shop flexible manufacturing systems. There are 3 kinds of production flexibility considered: (1) machine flexibility (MF), an operation can be performed on different machines; (2) operation flexibility (OF), different operations can perform the same job; (3) process flexibility (PF), no precedence constraints between operations. The optimality of scheduling depends on the result of process planning. From the IPPS data set, there are 3 kinds of information decided: Enterprise Resources: (1) Material information does not consider; (2) we only consider 2 kinds of resources, machine and shipment. Production (Job) Design: (4) multiple machines assignment to each operation is given (MF); (5) operation precedence constraint to each job is given (PF); (6) operation shipment constraint is given (OF); (7) operation time on the machine is given; Production Plan: (8) Lot-size of each job is equal to 1; (9) the objective is makespan. Decision variables are (10) precedence of operations, (11) machine assignments. Where shipment constraints are 0-1 variables, check the operations flexibility. All of IPPS descriptions are included in our model. In this experiment, we combine our hEA with priGA [18], rkPSO [31].

The resource assignment is given, and the complex resources assignments also are given. By using proposed hEA, random key-based section and state permutation sections are considered. We generated 24 IPPS Tests with different number of jobs. The evolutionary parameter’s initial settings are taken as shown in Table 1. Table 2 summarizes the experimental results. Although consider the small-scale problems, hEA has same performance with prGA and rkPSO, but hEA has a high performance for solving large scale IPPS problems.

<table>
<thead>
<tr>
<th>Table 1: Parameters And Strategies Of priGA, rkPSO, Proposed hEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Representation</td>
</tr>
<tr>
<td>Integer state</td>
</tr>
<tr>
<td>Selection</td>
</tr>
<tr>
<td>Clutchsize</td>
</tr>
<tr>
<td>Evol. Operations</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 2: Performance Comparisons with Different Approaches by 24 IPPS Tests

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Jobs</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>priGA</td>
<td>438.70</td>
<td>349.60</td>
<td>360.30</td>
<td>306.40</td>
<td>333.00</td>
<td>451.60</td>
<td>377.40</td>
<td>356.90</td>
<td>432.80</td>
<td>455.20</td>
<td>374.20</td>
<td>339.00</td>
</tr>
<tr>
<td>rkPSO</td>
<td>437.60</td>
<td>350.33</td>
<td>361.20</td>
<td>305.20</td>
<td>334.50</td>
<td>451.20</td>
<td>380.20</td>
<td>340.80</td>
<td>428.40</td>
<td>457.30</td>
<td>370.50</td>
<td>341.33</td>
</tr>
<tr>
<td>hEA</td>
<td>438.90</td>
<td>349.80</td>
<td>360.00</td>
<td>305.80</td>
<td>332.80</td>
<td>450.80</td>
<td>379.40</td>
<td>352.10</td>
<td>419.50</td>
<td>450.70</td>
<td>359.50</td>
<td>328.40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Jobs</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>priGA</td>
<td>455.40</td>
<td>395.50</td>
<td>444.60</td>
<td>472.10</td>
<td>437.50</td>
<td>391.70</td>
<td>447.40</td>
<td>440.50</td>
<td>482.60</td>
<td>521.80</td>
<td>497.00</td>
<td>564.80</td>
</tr>
<tr>
<td>rkPSO</td>
<td>450.33</td>
<td>398.20</td>
<td>440.50</td>
<td>468.72</td>
<td>435.50</td>
<td>385.66</td>
<td>440.50</td>
<td>435.50</td>
<td>473.40</td>
<td>513.60</td>
<td>487.50</td>
<td>560.40</td>
</tr>
<tr>
<td>hEA</td>
<td>436.80</td>
<td>378.80</td>
<td>424.20</td>
<td>447.60</td>
<td>412.50</td>
<td>370.40</td>
<td>410.80</td>
<td>420.00</td>
<td>454.33</td>
<td>494.50</td>
<td>451.50</td>
<td>520.60</td>
</tr>
</tbody>
</table>

5 CONCLUSION

This paper proposed a hybrid evolutionary algorithm (hEA) to solve IPPS problems. We proposed a network modeling way to formulate the IPPS problem into a unified model. This IPPS model specifies what manufacturing resources and operations/routes are needed to produce a product (a job). This unified IPPS model is formulated as a combinatorial optimization problem to decide operation sequence and resources assignment. For solving this unified IPPS model, we proposed a multi-section representation with combine a random key-based section and state permutation sections. We also considered 3 evolution policies: deletion policy with exploitation-based evolutionary operation; selection policy with roulette wheel selection; and clutchsize policy. Finally, the experiments verified the effectiveness of proposed algorithm, by comparing with different evolutionary approaches for the several test problems.

ACKNOWLEDGEMENTS

This work is partly supported by the Fundamental Research Funds (Software+X) of Dalian University of Technology (No. DUT12JR05), and also is partly supported by the Ministry of Education, Science and Culture, the Japanese Government: Grant-in-Aid for Scientific Research (C) (No.245102190001)

REFERENCES


295-15


